Academic Integrity For Computer Science Instructors

Thomas Lancaster

Department of Computing

Imperial College London

Abstract

For Computer Science instructors, upholding academic integrity requires approaching teaching and assessment in a way that communicates progressive principles such as honesty, trust, fairness, respect and responsibility to students, At the same time, instructors also have to take steps to make it untenable for students to commit academic misconduct. This means that instructors need to be aware of unacceptable conduct by students, covering behaviours such as plagiarism, collusion, contract cheating, examination cheating and research fraud. Instructors also need to put measures into place to design out opportunities for students to engage in such unacceptable behaviours.

This chapter explores academic integrity from the perspective of the knowledge needed by a Computer Science instructor. This is a changeable feast, as new methods to subvert academic integrity are always emerging, particularly in Computer Science where many students have the skills needed to develop the technology that aids new ways of cheating. As such, the chapter recommends that instructors deliver their curriculum with a pro-active focus on academic integrity from the outset. This includes leading by example and developing assessments that remove easy opportunities for students to cheat. This also means putting methods of detecting academic misconduct in place, even if detecting misconduct is only really intended as a measure designed to disincentivise students from cheating since they may get caught.

Keywords academic integrity, academic misconduct, plagiarism, contract cheating

Introduction

The term academic integrity describes the positive action where a student strives to take full advantage of the learning opportunities offered to them, complete assessment to the best of their ability and engage productively with the university community. It is part of the framework of ethical understanding that students are expected to gain during their journey to become a professional. Academic integrity is often presented alongside a parallel, yet less positive message that academic integrity means that students must not cheat. As such, the research and teaching practice on academic integrity lies closely intertwined with that related to student plagiarism.

For the Computer Science instructor, communicating and teaching the principles behind academic integrity offers challenge. Industry practices require students to gain experience collaborating and working in teams. Software development often involves the reuse of existing code bases. Students need to demonstrate that they are skilled at using online resources to find solutions to the problems they come across during the programming process. Students therefore need to understand the difference between acceptable and unacceptable conduct in a variety of educational and work-like situations.

The Computer Science instructor too must operate within the same academic integrity constraints as their students. This means that they must lead by example, taking every opportunity to design learning resources that remove opportunities for students to accidentally breach academic integrity norms. This also means that they should ensure that, where students do commit plagiarism or fail to follow acceptable academic integrity practice, suitable action is taken.

This chapter addresses academic integrity for the Computer Science discipline by providing practical ideas and considerations. After a brief introduction to academic integrity terminology and challenges, the chapter focuses on three specific problems: (1) introducing the importance of academic integrity to students; (2) engaging students with assessment and reducing the temptation for them to cheat; and (3) detecting when academic integrity has been breached. Detection, although somewhat counter to presenting academic integrity as a set of positive virtues, remains a necessary evil since it is not fair for some students to gain qualifications they do not deserve at the expense of other students who are putting the expected effort into their studies.

This chapter is intended to be of use to all Computer Science instructors looking to improve how they ensure academic integrity is upheld within their teaching. It is aimed at instructors of all experience levels. It is also hoped that the chapter will be of use beyond the Computer Science discipline, particularly for other practical and industry led subjects.

No attempt is made in this chapter to provide a source for every idea. It includes, for example, many now standard observations and recommendations on assessment design. These have been widely mentioned in the literature, including in other papers by the author and it would be difficult to credit these to a single definitive

source. Instead, it is intended to collate the relevant information in a succinct form suitable as a primer for the modern Computer Science instructor.

Academic Integrity in Computer Science

The Exemplary Academic Integrity Project (2013) defines academic integrity as:

> "acting with the values of honesty, trust, fairness, respect and responsibility in learning, teaching and research. It is important for students, teachers, researchers and professional staff to act in an honest way, be responsible for their actions, and show fairness in every part of their work. All students and staff should be an example to others of how to act with integrity in their study and work. Academic integrity is important for an individual's and a school's reputation."

The definition is useful as it reverses the traditional way in which academic integrity is presented, which is simply by telling a student what they may not do. The definitions provides a set of values for students to follow during their academic life. It further notes that these values apply equally to everyone involved in education, which also includes instructors.

Despite an attempt to focus on the positive, any practical chapter on academic integrity would be incomplete without indicating the unacceptable behaviours that students have been observed using. An overarching term for such behaviours is academic misconduct. These behaviours are also often referred to simply as methods of student cheating.

Several of the methods through which academic integrity can be breached and which are most relevant for Computer Science are summarised in Table 1.

Academic Integrity Breach	Description
Plagiarism	The process where a student uses the words or ideas of another without acknowledgement. A particular issue in Computer Science is *source code plagiarism*, where program code is copied or reused in an unacceptable manner. A related area is *essay spinning*, where students use automated software, such as that intended to translate between languages, to disguise plagiarised work and avoid detection (Lancaster and Clarke 2009; Jones and Sheridan 2015).
Collusion	Where two or more students have worked closely together during the production of assessed work, going beyond acceptable levels of collaboration. When collusion is taken to the level where the submitted work of two students is identical or very similar, this also represents a form of plagiarism.
Contract Cheating	The behaviour where a student uses a third party to complete assessed work for them, or attempts to do use a third party in this manner (Clarke and Lancaster 2006). This involves an exchange of benefits for the parties involved, often in the form of a fee. Contract cheating was originally observed primarily relating to

	the outsourcing of technical Computer Science assignments, including programming, but also covers written work, such as reports. Some contract cheating is facilitated through *essay mills*, companies that are primarily online and which offer to produce original bespoke assessments for students.
Exam Cheating	Where a student attempts to gain an unfair advantage in an examination, for instance by referring to concealed notes or through secret communication with an individual outside an exam hall. This behaviour includes *impersonation*, where a third party attempts to switch places with an individual who is meant to be there so that they can sit the exam for them. This can be possible for exams taken face-to-face or online. One enabling mechanism for impersonation includes having a test taker disguised to look like the person meant to be taking the exam. A second enabler involves student identification cards being doctored so that the third party does not raise suspicion.
Research Fraud	The process where research results and conclusions are returned that are not backed up by verifiable evidence. This can include deliberate errors in the research methodology, such as mistreating or ignoring data. This can also include not acting within a defined ethical framework. A specific example is *data fabrication*, where reliable data is not collected but instead produced in a way that benefits the student in achieving the result or conclusion they would like to see. In Computer Science, this may perhaps be an issue during the Project stage of a student's course.

Table 1. Types of academic integrity breaches of interest to the Computer Science instructor

The definitions in Table 1 are not intended to offer a complete list of the ways in which academic misconduct can be breached. Such a list can never be complete so long as new methods of teaching and assessment are developed and these are accompanied by revised methods of cheating. Academic misconduct can include any activity intended to give a student an unfair advantage over other people. Other examples include students using social engineering techniques to gain access to early copies of exam papers. They can include students hacking into computer systems to change their marks. There are also areas where instructor opinions of what constitutes academic misconduct may vary, such as where a student uses chemical enhancements, such as study drugs, designed to allow them to concentrate for longer and hence gain a higher mark than their peers.

There are also areas where academic integrity can be breached, but this may not be deliberate. For example, a student could be judged to have plagiarised as a result of poor study skills. They could have committed research fraud through misunderstanding research methodologies or incorrectly processing data. This is why education and support for students is important. Addressing this once an academic integrity breach has taken place is too late. This is because it is not always possible to tell if this was intentional or accidental.

A suitable understanding of what plagiarism is and why this is considered a problem will help instructors trying to extend their knowledge to other forms for academic misconduct. Many commentators have linked a growth in academic misconduct with the prevalence of Internet aided plagiarism (Austin and Brown 1999). Students entering higher education have grown up surrounded with the Internet for all of their life and so their concept of the value and ownership of information may be different to that of their instructors.

Computer Science instructors also have to be aware of the types of academic misconduct that appear to be becoming more visible. For example, research into contract cheating has regularly seen assessments from across the Computer Science spectrum outsourced (Jenkins and Helmore 2006; Lancaster and Clarke 2007). Technical assignments, particularly those which do not require English language proficiency, can be contracted out to a global marketplace, often for a cost that is economically feasible for students. The range of assessments that can be outsourced contains activities contributing strongly to a student's final degree classification. This includes major work such as a final year project, assessments completed in stages and tasks where students are intended to provide their own personal reflection.

The use of examinations may remove some of the academic integrity challenges seen with coursework, but examinations themselves can be susceptible to contract cheating. Students have been observed paying or using a third party to provide them with undocumented help in an exam situation (Lancaster and Clarke, 2017). There are also examples where students have used an impersonator to take an exam for them. With the move for Computer Science courses to be delivered online, it appears that such courses have particular vulnerabilities where academic integrity can be breached. That means that special attention has to be paid to the design and delivery of such online courses.

Teaching Academic Integrity Principles

Students need to be equipped with the necessary tools for success as part of their Computer Science course. This includes helping students to develop an understanding of how they can progress through their studies whilst always holding academic integrity in their mind. For the instructor, this requires supporting students to avoid accidentally breaching academic integrity principles.

It is unlikely to be sufficient for an instructor to assume that students arriving at university all share a common understanding of what academic integrity is, why it is important or how they can act with integrity. This can particularly be the case where students arrive from different cultures, educational upbringings and backgrounds.

This means that teaching academic integrity is necessary. Such teaching needs to cover two main considerations.

First, students need to understand what academic integrity means, why this is essential to their studies and how academic integrity values relate to their everyday life.

Second, students need to be shown the practical techniques they will require during their course. These are the techniques necessary for them to avoid academic integrity breaches and to show that they respect the resources that they are working with.

As with many subjects, academic integrity is not a subject that can just be taught once and forgotten about. One model that can be used is to introduce students to the appropriate concepts early on in their course, integrating opportunities for students to receive formative feedback on how well they have engaged with the material. These concepts can then be developed and reinforced throughout the course, with students provided with information specific to the subject they are working on and their level to study.

For example, when arriving at university, students may need to be shown the practical skills of finding and referencing textual information very early on. This teaching would be supported by exploring why valuing the ownership of information is important and the possible consequences that taking information from a source without acknowledgement would lead to in their everyday life. In a career situation, this could lead to an employee being dismissed, Intellectual property breaches could also lead to litigation for the company they work for.

Discussions about other types of academic misconduct and how this can be breached can occur at appropriate points during the student journey. For instance, correct examination conduct and the skills needed to be successful with exams can be explored early on in their course before students take low risk tests. In Computer Science, presentation of the concept of examination success can also be related to the professional examinations, such as vendor certifications. These examinations are ones that students may be expected to take beyond their time at university and throughout their career.

Introducing a discussion on contract cheating can be more difficult. Some instructors may believe that all students already know this is wrong, but such a view does not account for differences in student cultural upbringing. There are cultures where students are taught that using the words of experts is not only allowable, it is expected. There are also students who already know that this is wrong and are still contract cheating. The counter view held by some students, that information belongs to them once they have paid for or acquired it, needs to be openly explored.

Challenging conversations can be framed as part of a wider discussion regarding what level of external help is permissible. This is an important distinction when attempting to build Computer Science student team working skills ready for industry, but still ensuring that students receive credit for their own individual effort and contributions.

Many academic integrity discussions also need to happen at the level of individual subjects. A student learning introductory programming skills does need to demonstrate that they have mastered basic coding concepts, even though the level

of exercises set may be trivial to many experienced programmers. This means that taking a code fragment from an existing online codebase to solve a simple exercise regarding manipulating strings, for example, would not be considered appropriate at such an early stage of the student's academic journey. Once a student is required to demonstrate that they are ready for a professional programming career, the expectations on them may be different. In such a case, it may be considered both acceptable and desirable for a student to demonstrate that they can find and reuse existing code fragments.

There may be further complications when a student needs a resource to complete an assessment, but that resource itself is not a key area that is being examined. An example may be on a web development assessment, where a student wishes to improve the look of their website by using existing artwork, but it is their coding skills that are being assessed. In such a case, students need to be made aware of the instructor's expectations of how they demonstrate academic integrity.

For Computer Science, Simon et al (2016) recommend that exactly what students are allowed to do and what they may not do is spelled out for them on the assignment brief. This may include detailing which individuals and services they can and cannot get help from, which resources they may and may not use and what aspects of the assessment they can and cannot solicit help with. It may also be worth agreeing such a list with students as part of the assessment development regime, thus also helping the student cohort to feel some ownership of the decisions made.

One message that is worth communicating to students is that academic misconduct is not a victimless crime. A student who used dishonest tactics to get ahead could end up getting better marks than a student that worked hard. A cheating student could even get a job that they don't deserve, taking this away from a more honest student. That is why it's important for instructors and students to work in close partnership and develop a shared understanding of why academic integrity is important for all parties operating within the higher education landscape.

Assessing with Academic Integrity

Both students and instructors have a role to play in ensuring that academic integrity is maintained through the assessment process. As this chapter has already discussed, students need to understand the benefits of completing their own work and commit to doing this to the best of their ability. Students also need to be equipped with the academic skills necessary for their own success.

Instructors have further responsibilities during the assessment process, an area which students often consider the most important part of their educational experience. Instructors need to develop assessments that reduce the opportunity for students to accidentally breach academic integrity. They should also try and remove any of the temptations that may mean that some students would choose to cheat if

they thought that they could get away with it. This can be accomplished by rethinking and restructuring the assessment process, or by removing any benefits that attempting to cheat may bring.

Student interest in assessment can be improved by making this more engaging. Mechanisms for doing this will vary between student cohorts, so some partnership working with students is necessary here. It may, for example, be that a particular student group prefers examinations over coursework, group assessments over individual assessments or practical tasks over written ones. It may be possible to co-design assessments with students so that they feel some ownership of the assessments tasks that are set.

There may also be cases where students feel that assessments are simply hurdles that they need to jump. Working with students can help to share the message that these are measures of progress. Demonstrating to students that formative assessment is useful can also pay dividends, particularly in the areas of Computer Science where students need to build up skills gradually, such as when learning computer programming.

Assessments can be developed so that students cannot gain a passing mark by using repeatedly cheating using the same techniques. A simple way of doing this would be to set two different types of assessment within a subject, such as both a coursework and a test. In such a case, the students would be required to demonstrate their competency in both items. In this particular circumstance, a student who had plagiarised their coursework and not been caught would still have to find a different way to cheat during the supervised test element. As well as engaging the students using multiple assessment modalities, this also reduces the benefit of cheating, since the required learning would still need to be completed for the second element of assessment.

A supervised component of coursework assessment can also be useful, particularly where instructors are worried about contract cheating (Lancaster and Clarke 2016). This can include both written tests and practical tests, but also supervised coursework sessions, presentations, spoken viva voce examinations, industry style coding interviews and product demonstrations. There are many opportunities to innovate here.

Employability initiatives are also worthy of consideration. For instance, more authentic forms of teaching and assessment can be used, such as a simulated work environment where students work on assigned projects during office hours to develop and document a solution to a problem posed by an external client. Such assessments can also be of value for students when they looking to present a professional portfolio and to showcase on their Curriculum Vitae that they have acquired a wide range of new skills during their course.

Instructors do need to be wary about the need to balance several trade-offs when developing assessments. They have to balance encouraging academic integrity, reducing the value of academic misconduct, meeting industry requirements and their own ability to manage the assessment process within their wider workload commitments. As an example, an individual spoken examination, or viva voce, may be a

very good method of authenticating that a student knows and understand a subject. Despite the advantage the time commitment necessary to do this fairly for all students in a large Computer Science class may mean that a spoken exam is simply not practical. Even if the time required is not the issue, there are still issues of fairness to consider, such as how the instructor ensures that students who have their viva early in the process are not at a disadvantage over students scheduled later on. The latter group could be said to have both more time to prepare and the potential to quiz earlier students over the questions that they are likely to be asked. There can also be challenges posed to the consistency of processes when a large team of assessors is used.

Compromises are possible and recommended. It may be that instructors work together to ensure a varied diet of assessments across the whole year of a course, with extra academic integrity measures put into place during those assessments that underpin later learning. For instance, the crucial assessments in the first year of a traditional Computer Science course might be considered to be those for the core programming and mathematics subjects, an understanding of which is needed for success in later years. It is also possible to set an end-of-year viva, bringing together the skills from across a variety of subjects and helping to check that students have an understanding of how these distinct areas all integrate together. Such a viva can also be resourced using multiple instructors to ensure that this process is manageable. Major modules, such as a final year student project that often represents the culmination of a degree, can also have extra academic integrity checkpoints put into place.

Another option for assessment that combines several recommendations of good practice is for an instructor to structure this using multiple linked components. Here is a simple example regarding how this could be used for a computer programming module, but similar approaches are also possible for other subjects. First, the student completes a coursework assessment as usual, where they develop a solution to a programming problem. They submit their code for that coursework. At a scheduled time, they attend a practical examination where they are required to make changes to the code they already submitted. This approach requires students to be familiar with their code, reducing the usefulness of them having outsourced its production or used other cheating methods. Where students chose not to engage with academic integrity during the coursework process they would still have to master the programming skills required for the practical exam. The student should realise that there is little benefit to them not putting in the effort during the coursework phase, hence encouraging them to concentrate properly on the assessment.

Most of the examples in this part of the chapter have focused on coursework assessment, but examinations are also susceptible to cheating. Even an activity like a viva voce examination can be cheated on if a student has advance access to a set of standard questions, or if someone outside the exam room can feed the student answers through a concealed earpiece. Some examination practices used by instructors are themselves questionable, for instance where the questions have been taken from previous papers that students have access to, or match those provided in a

sample paper. This would make the exam merely test of memory skills and not ability. Hence, instructors need to be continually aware of opportunities where their own practice can lead to the manifestation of academic misconduct.

Academic integrity in examinations can be further preserved by carefully developed examination processes. This includes specifying the equipment that students are allowed to use, supplying equipment they cannot have tampered with such as pens and calculators, monitoring and recording screen activity during practical examinations, and ensuring that all examination questions are original and go through a robust quality checking and moderation process. If nothing else, all examinations need to be invigilated and students alert to the fact that their actions are being watched, thus removing the benefit from any opportunistic attempts by students to cheat.

Detecting Breaches of Academic Integrity

Instructors need to be aware that detection tools and punitive measures are not a single solution to academic integrity challenges, but they can be used to support other approaches. Tools can also provide instructors with some assurance where they believe that academic integrity has been breached. Many software tools already exist for the detection of breaches of academic integrity and the choice of the most suitable tool has to be closely matched to different situations. But software cannot be considered a complete or infallible solution. It may always possible for students to cheat in ways that tools do not detect. Many web pages, videos and social media posts exist where students share the latest ideas to defeat detection technology.

Tools for detecting traditional forms of plagiarism are available and this is a field that is well-established. These tools are available to identify similarity in written text, within source code submissions in many programming languages and for specific technical environments such as spreadsheet assignments or database assignments. These tools flag student submissions that warrant further manual investigation by an instructor. For instance, this may include a written student submission where sections of the work match Internet sources. The results from the tools do not directly indicate plagiarism, so human judgement is required. There are many acceptable reasons why a tool may give such a false hit, for instance two textual documents may appear to match each other, but the cause may turn out to be the use of a correctly cited quotation that was common to both documents.

Although plagiarism detection tools are well-established, tools designed to detect contract cheating are not so readily available. Indications are that students are turning to third party writers, programmers and other contractors precisely because the original work they supply is unlikely to be detected using current plagiarism detection technology.

The best advice available at present is for instructors to approach assessing coursework with a keen eye and questioning mind, with the view that that someone

other than the student may have produced this item of coursework. For an instructor who knows a student, their abilities or their writing style, such simple checks as looking at the properties of the document they submitted can be useful. These may show suspicious file creation dates or author names.

Several techniques that could be used to support the automated detection of contract cheating have also shown promise. Checks that the writing style of a student remains consistent from one assessment to the next are possible. Some work has been made to automate such checks (Juola 2017). Similar techniques could be used to decide if student programming style remains consistent from one exercise to the next. It has been suggested that contextual information could be extracted from student submissions to help develop artificial intelligence systems to detect contract cheating (Lancaster and Clarke 2014). Student mark profiles can also be analysed to see where student performance shows inconsistencies of the type that may suggest that the student has received external help (Clare et al 2017).

The use of tools to detect academic misconduct is an important part of any complete academic integrity process. Using tools protects the value of the results and overall qualifications awarded to the students who are working with academic integrity. Their use shows that academic integrity is being treated seriously. The existence of such tools and evidence of their use also provides a deterrent effect to students who may otherwise have considered cheating, but have changed that view due to the risk that they may be caught.

If an investigation suggests that a student has breached academic integrity regulations, it is important that a fair, transparent and consistent approach is followed. This means that all students in this situation are considered using an identical process and treated in the same way. Doing this ensures that the wider higher education setting also complies with the underlying principles of academic integrity. Where a fair process does identify that a student has breached academic integrity, this also has to lead to a suitable outcome, such as a fairly and consistently applied penalty for the student. There may also be the need for arrangements to support the student in future to be put into place.

The same technology used to detect plagiarism and potentially penalise students can also be used in a positive manner. For example, plagiarism detection software can be introduced in a formative way to allow students to find out if they are accidentally demonstrating poor academic practice (Halgamuge 2017). As this is formative, the process can then be used to put support into place for students, rather than attempting to penalise them. For instance, where a student draft report is run through plagiarism detection software and found to contain copied text, the student could be provided with support on how they should correctly acknowledge and reference information from an external source. Such support is useful for students to ensure that they do not accidentally breach academic processes when in a summative situation.

Conclusions

An understanding of academic integrity, including how academic integrity can be promoted to students and how academic integrity should be continually maintained, is important knowledge needed by all current Computer Science instructors. This chapter has motivated the need for academic integrity in Computer Science education and has provided an overview of current thinking and ideas within the field.

When students cheat, many will have considered that they are taking a risk in return for a potential reward. For such a student, cheating has to offer them some sort of benefit, such as passing an assessment point or obtaining a better mark than they otherwise would have done. That benefit has to outweigh the chance of them being caught and the likely knowledge of what will happen to them if they are found to have breached academic integrity.

Some students cheat because they do not know any better. That risk can be mitigated against by ensuring that a programme of academic integrity tuition and support is a core component of study for all students.

Other students cheat because they fear failure and believe that their academic misconduct is unlikely to be detected. That risk to integrity can be reduced through careful instructional design and considered assessment processes.

There are students who cheat because they do not see the value in what they are being taught or the assessments they are being asked to do. This risk can also be mitigated against by developing engaging teaching and incorporating real world examples.

In Computer Science, there are also students who have the ability to complete assessment for themselves, but just want to show that they can beat the system. This is an example of the long-standing hacker mentality already evident in the Computer Science population. There is no single solution to prevent this, but here simply ensuring that students are kept engaged and interested may offer the key.

Academic integrity is not presented as just a problem to be solved. It may be impossible to completely ensure it, particularly when students are technically sophisticated and the tools they have available to help them to take shortcuts to succeed are always changing. After all, students are using methods of cheating now that were unheard of a decade ago. But there are also great opportunities for the Computer Science academic. Those academics are well-equipped to be at the forefront of future academic integrity research. Such academics are more likely than most to have the technical skills necessary to conduct research into ongoing topics as plagiarism detection or emerging topics such as the detection of contract cheating or essay spinning.

Academic integrity within the Computer Science discipline can also be supported by using a wide variety of assessment types, including work simulations. Not every career path has the same view of integrity, even though documents such as professional body authorised codes of conduct offer some general principles.

Within Computer Science education, there are areas where the best solution towards ensuring academic integrity is not clear, such as where the border between unacceptable collusion and expected workplace collaboration is concerned.

The opportunities for Computer Science academics to develop processes, definitions and solutions for the preservation of academic honesty also offers opportunities for these academics. Many academic integrity challenges, such as those linking employability and collaboration, are likely to be more prevalent in Computer Science than in other academic disciplines. This also means that the prospect exists for the Computer Science specialist to take a leadership role in the academic integrity field.

Above all, academic integrity needs to be more than just an add-on to Computer Science. Many of the same principles used to design good assessments also reduce the opportunities for academic misconduct. They are already designed to engage students and to help them to succeed. An instructor working in this manner is therefore operating with many of the most important principles of academic integrity at heart.

References

Austin, M. and Brown, L., 1999. Internet plagiarism: developing strategies to curb student academic dishonesty, The Internet and Higher Education., 2(1), pp. 21-33.

Clare, J., Walker, S. and Hobson, J., 2017. Can we detect contract cheating using existing assessment data? Applying crime prevention theory to an academic integrity issue. International Journal for Educational Integrity, 13(1).

Clarke, R. and Lancaster, T., 2006. Eliminating the successor to plagiarism? Identifying the usage of contract cheating sites. 2nd Plagiarism: Prevention, Practice and Policy Conference 2006, Newcastle, United Kingdom.

Exemplary Academic Integrity Project, 2013, Resources on academic integrity. https://lo.unisa.edu.au/course/view.php?id=6751&section=6 [Accessed 18 April 2018].

Halgamuge, M., 2017. The use and analysis of anti-plagiarism software: Turnitin tool for formative assessment and feedback. Computer Applications in Engineering Education.

Jenkins, T. and Helmore, S., 2006. Coursework for cash: the threat from online plagiarism. Proceedings of 7th Annual Higher Education Academy Conference in Information and Computer Sciences. Dublin, Ireland.

Jones, M. and Sheridan, L., 2015. Back translation: an emerging sophisticated cyber strategy to subvert advances in 'digital age' plagiarism detection and prevention. Assessment & Evaluation in Higher Education, 40(5), pp.712-724.

Juola, 2017. Detecting contract cheating via stylometric methods, Plagiarism across Europe and Beyond 2017, Brno, Czech Republic.

Lancaster, T. and Clarke, R., 2007. Assessing contract cheating through auction sites – a computing perspective; 8th Annual Higher Education Academy Conference in Information and Computer Sciences, Southampton, United Kingdom.

Lancaster, T. and Clarke, R., 2009. Automated essay spinning – an initial investigation; 10th Annual Higher Education Academy Conference in Information and Computer Sciences, Kent, United Kingdom.

Lancaster, T. and Clarke, R., 2014. An initial analysis of the contextual information available within auction posts on contract cheating agency websites, Proceedings of International Workshop on Informatics for Intelligent Context-Aware Enterprise Systems (ICAES 2014), University of Victoria, Victoria, Canada.

Lancaster, T. and Clarke, R., 2016. Contract cheating – the outsourcing of assessed student work, in Handbook of Academic Integrity, Bretag, T. (editor): SpringerReference.

Lancaster, T. and Clarke, R., 2017. Rethinking Assessment By Examination In The Age Of Contract Cheating, Plagiarism Across Europe and Beyond 2017, Brno, Czech Republic.

Simon, Sheard, J., Morgan, M., Petersen, A., Settle, A., Sinclair, J., Cross, G. and Riedesel, C., 2016. Negotiating the maze of academic integrity in computing education. Proceedings of the 2016 ITiCSE Working Group Reports, pp. 57-80.